# OASIS ebXML Messaging Services Version 3.0: Securing Messages with SAML Tokens under the WS-Security Profile

Version

1.3 Draft

Date

 Thursday, 27 May 2013

Contributors:

|  |  |
| --- | --- |
| Ian Otto | Australian Government Department of Industry, Innovation, Climate Change, Science, Research, and Tertiary Education |
| Malcolm Young | Australian Government Department of Industry, Innovation, Climate Change, Science, Research, and Tertiary Education |
| Michael Leditschke | Australian Government Department of the Treasury |

Table of Contents

[Securing EBMS3 with SAML Tokens under the WS-Security Profile 1](#_Toc351553262)

[1. Introduction 2](#_Toc351553263)

[1.1. Background and Objectives 2](#_Toc351553264)

[1.2. Scope 2](#_Toc351553265)

[1.3. Federated Identity Providers and Their Role in an eBusiness Messaging Framework 2](#_Toc351553266)

[1.4. Caveats and Assumptions 2](#_Toc351553267)

[1.5. General Rules for Normative Interpretation 3](#_Toc351553268)

[1.6. XML Notation 3](#_Toc351553269)

[1.7. Example Domains 3](#_Toc351553270)

[1.8. Normative References 3](#_Toc351553271)

[2. SAML Tokens For ebMS 6](#_Toc351553272)

[2.1. SAML Subject and Attributes 6](#_Toc351553273)

[2.2. Types of SAML Tokens 6](#_Toc351553274)

[2.3. Proof Key Applicability to Multi-Hop Scenarios 6](#_Toc351553275)

[2.4. AppliesTo, Audience, and SubjectConfirmation 6](#_Toc351553276)

[3. Processing Modes for SAML Security 8](#_Toc351553277)

# Introduction

This specification describes the applicability and usage of SAML Assertions (obtained through WS-Trust) along-side Username Token and X.509 Token in securing ebXML Messaging Service (ebMS) messaging. SAML Assertions provide a more dynamic way of establishing and managing identity and roles than their counter-parts.

## Background and Objectives

[EBMS3CORE] states that WS-Security is the mechanism that is used to secure messages, and then goes on to provide examples for X.509 Tokens and Username Tokens in specific can be used. The use of SAML is alluded to but not specified.

The purpose of this specification is to provide insight and guidance on how SAML should be used in the ebMS context.

The core SAML specifications cover two facets of SAML, SAML Assertions which are a representation of an identity, and the SAML Protocol which is a way that SAML Assertions can be exchanged to achieve various goals. SAML Assertions have a wide applicability outside of the SAML Protocol and it is this usage which is important in the specification. The SAML Protocol will not be discussed further in this document.

The versions of SAML in common use are SAML 1.1 and SAML 2.0. While syntactically different, the two specifications are semantically similar enough that they can be treated as being the same for the purposes of this specification.

## Scope

The document will describe the significant points on how SAML tokens may be used in the securing of ebMS3 messages.

## Federated Identity Providers and Their Role in an eBusiness Messaging Framework

By and large, ebMS is used to exchange messages in communities where there is a relatively stable membership. The cost of changing passwords occasionally or rolling of X.509 credentials as they expire can be effectively managed albeit with some cost.

ebMS however is starting to be used in communities where there is a large number of clients talking to one or more hubs. In such communities, use of an Identity Provider can reduce the burden on both the client and the hub.

There are a numerous benefits including:

* when the hub needs to roll its X.509 credentials, it only has to notify the Identity Provider, not each of the clients.
* identity providers isolate the hub from technology changes. Regardless of how the client authenticates to the identity provider, the hub will receive identity as a standard SAML Assertion.
* the identity information may be supplemented by additional information, for instance for authorization purposes.

## Caveats and Assumptions

The target audience for this specification is the community of software developers who will implement the ebXML Messaging Service.

It is assumed the reader has an understanding of communications protocols, MIME, XML, SOAP, SOAP Messages with Attachments and security technologies.

All examples are to be considered non-normative. If inconsistencies exist between the specification and the examples, the specification supersedes the examples.

## General Rules for Normative Interpretation

The key words *MUST*, *MUST NOT*, *REQUIRED*, *SHALL*, *SHALL NOT*, *SHOULD*, *SHOULD NOT*, *RECOMMENDED*, *MAY*, and *OPTIONAL* in this document are to be interpreted as described in [RFC2119].

For any given module described in this specification, an implementation MUST satisfy ALL of the following conditions to be considered a conforming implementation of that module:

1. It supports all the mandatory syntax, features and behavior (as identified by the [RFC2119] key words MUST, MUST NOT, REQUIRED, SHALL and SHALL NOT) defined in the section that specifies that module.
2. When the keywords MUST, SHALL, or REQUIRED are used to qualify a feature, support for this feature--either message content or implementation behavior--is mandatory in an implementation with a conformance profile that requires this feature.
3. It complies with the following interpretation of the keywords OPTIONAL and MAY: When these keywords apply to the behavior of the implementation, the implementation is free to support these behaviors or not, as meant in [RFC2119]. When these keywords apply to message contents relevant to a module of features, a conforming implementation of such a module MUST be capable of processing these optional message contents according to the described ebXML semantics.
4. If it has implemented optional syntax, features and/or behavior defined in this specification, it MUST be capable of interoperating with another implementation that has not implemented the optional syntax, features and/or behavior. It MUST be capable of processing the prescribed failure mechanism for those optional features it has chosen to implement.
5. It is capable of interoperating with another implementation that has chosen to implement optional syntax, features and/or behavior, defined in this specification, it has chosen not to implement. Handling of unsupported features SHALL be implemented in accordance with the prescribed failure mechanism defined for the feature.

## XML Notation

When describing concrete XML schemas and information items, this specification uses a convention in which each XML element or attribute is identified using abbreviated [XPATH] notation (e.g., /x:MyHeader/x:SomeProperty/@attribute).

## Example Domains

Hostnames used in the examples are fictitious, and conform to [RFC2606]. The example.org domain is intended to refer generically to a relevant industry standards organization, while the example.com domain represents a participant in a message exchange (whether commercial, government, or other entity).

## Normative References

[EBMS3CORE] OASIS Standard, OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features, October 2007,
<http://www.oasisopen.org/committees/download.php/24618/ebms\_core-3.0-spec-cs-02.pdf>

[SAMLCoreV1] Oasis Standard, E. Maler, P.Mishra, and R. Philpott (Editors), Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1, September 2003.
< http://www.oasis-open.org/committees/documents.php?wg\_abbrev=security>

 [SAMLCoreV2] Oasis Standard, S. Cantor, J. Kemp, R. Philpott, E. Maler (Editors), Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0,
March 2005.
<http://docs.oasis-open.org/security/saml/v2.0/>

[WSSSAML] OASIS Standard, Kelvin Lawrence, Chris Kaler (Editors), Web Services Security:SAML Token Profile 1.1, 1 February 2006
<http://docs.oasis-open.org/wss/oasis-wss-SAMLTokenProfile-1.1>

[HTTP11] R. Fielding, et al, Hypertext Transfer Protocol -- HTTP/1.1, 1999. <http://www.ietf.org/rfc/rfc2616.txt>

[IANAMEDIA] Various, MIME Media Types, Various. <http://www.iana.org/assignments/media-types/>

[RFC2045] N Freed, et al, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, 1996. <http://www.ietf.org/rfc/rfc2045.txt>

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, 1997. <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2387] E. Levinson, The MIME Multipart/Related Content-type, 1998. <http://www.ietf.org/rfc/rfc2387.txt>

[RFC2392] E. Levinson, Content-ID and Message-ID Uniform Resource Locators, 1998. <http://www.ietf.org/rfc/rfc2392.txt>

[RFC2396] T. Berners-Lee, et al, Uniform Resource Identifiers (URI): Generic Syntax, 1998. <http://www.ietf.org/rfc/rfc2396.txt>

[RFC2822] P. Resnick, ed., Internet Message Format, 2001. <http://www.ietf.org/rfc/rfc2822.txt>

[SMTP] J. Klensin, ed., Simple Mail Transfer Protocol, 2001. <http://www.ietf.org/rfc/rfc2821.txt>

[SOAP11] D. Box, et al, Simple Object Access Protocol (SOAP) 1.1, 2000. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

[SOAP12] M. Gudgin, et al, SOAP Version 1.2 Part 1: Messaging Framework, 2003. <http://www.w3.org/TR/soap12-part1/>

[SOAPATTACH] J. Barton, et al, SOAP Messages with Attachments, 2000. <http://www.w3.org/TR/SOAP-attachments>

[UTF8] F. Yergeau, UTF-8, a transformation format of ISO 10646, 1998. <http://www.ietf.org/rfc/rfc2279.txt>

[WSIAP10] Chris Ferris, et al, eds, Attachments Profile Version 1.0, 2004. <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html>

[WSIBSP10] Abbie Barbir, et al, eds, Basic Security Profile Version 1.0, 2005. <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

[WSR11] Kazunori Iwasa, et al, eds, WS-Reliability 1.1, 2004. <http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws\_reliability-1.1-spec-os.pdf>

[WSRM11] D. Davis, et al, eds, Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1, 2007. <http://docs.oasis-open.org/ws-rx/wsrm/v1.1/wsrm.pdf>

[WSRMP11] D. Davis, et al, eds, Web Services Reliable Messaging Policy (WS-RM Policy) Version 1.1, 2007. <http://docs.oasis-open.org/ws-rx/wsrmp/v1.1/wsrmp.pdf>

[WSS10] Anthony Nadalin, et al, eds., Web Services Security: SOAP Message Security 1.0, 2004. <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

[WSS10-USER] P. Hallam-Baker, et al, eds., Web Services Security UsernameToken Profile 1.0, 2004. <http://docs.oasis-open.org/wss/2004/01/>

[WSS10-X509] P. Hallam-Baker, et al, eds., Web Services Security X.509 Certificate Token Profile, 2004. <http://docs.oasis-open.org/wss/2004/01/>

[WSS11] Anthony Nadalin, et al, eds., Web Services Security: SOAP Message Security 1.1, 2005. <http://docs.oasis-open.org/wss/v1.1/>

[WSS11-USER] A. Nadalin, et al, eds., Web Services Security UsernameToken Profile 1.1, 2006. <http://docs.oasis-open.org/wss/v1.1/>

[WSS11-X509] A. Nadalin, et al, eds., Web Services Security X.509 Certificate Token Profile 1.1, 2006. <http://docs.oasis-open.org/wss/v1.1/>

[XML10] Tim Bray, et al, eds., Extensible Markup Language (XML) 1.0 (Third Edition), 2004. <http://www.w3.org/TR/2004/REC-xml-20040204/>

[XMLDSIG] Donald Eastlake, et al, eds, XML-Signature Syntax and Processing, 2002. <http://www.w3.org/TR/xmldsig-core/>

[XMLENC] D. Eastlake, et al, XML Encryption Syntax and Processing, 2002. <http://www.w3.org/TR/xmlenc-core/>

[XMLNS] Tim Bray, et al, eds, Namespaces in XML, 1999. <http://www.w3.org/TR/REC-xml-names/>

[XMLSCHEMA] Henry S. Thompson, et al, eds., XML Schema Part 1: Structures Second Edition, 2004. <http://www.w3.org/TR/xmlschema-1/>

[XPATH] James Clark, et al, eds., XML Path Language (XPath) Version 1.0, 1999. <http://www.w3.org/TR/xpath>

## Non-Normative References

[WSPOLICY] A. Vedamuthu, et al, eds, *Web Services Policy 1.5: Framework*, 2007. <http://www.w3.org/TR/ws-policy/>

[WSSECPOL] A. Nadalin, et al, eds, *WS-SecurityPolicy 1.2*, 2007. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf>

# SAML Tokens For ebMS

A SAML Token (SAML Assertion) can be used to convey identity in a number of different circumstances including Web Services Security. [WSSSAML] defines the ways in which a SAML Token can be employed to secure Web Services.

Implementations MUST support SAML 2.0, although asSAML 1.1is still in wide use, implementations MAY choose to accept either token type as defined in [SAMLCoreV1] and [SAMLCoreV2].

## SAML Subject and Attributes

A SAML Subject identifies the initiating party that requested the Assertion. The SAML Subject is a unique and persistent identifier.

SAML Attributes (in WS-Trust parlance these are referred to as Claims) are attributes of the identity. An Assertion will contain zero or more Attributes that describe the subject.

The values of these SAML Attributes MAY be used as referred to in [EBMS3CORE] (7.12.7. Persistent Authorization) to authorize access.

## Types of SAML Tokens

SAML Assertions can be issued as Holder-Of-Key, Sender-Vouches, or Bearer tokens. Bearer tokens are not supported by the [WSSSAML]. Sender-Vouches tokens may be useful in some multi-hop scenarios, but are outside the scope of this version of the specification. (In Sender-Vouches, the receiver must trust the sender that they are authorised to use the attached SAML token. It is not cryptographically bound to them in any way.)

A Holder-Of-Key token contains a proof key that can be used to verify the originator of a message is the legitimate owner of the token.

If the proof key is a symmetric key, then it will be wrapped (using the public key of the recipient) for the intended recipient of the token. Only the recipient can validate the message.

If the proof key is the public half of an asymmetric key pair, then it can be include in the assertion unencrypted. In this case, anyone can validate a signature made by the message initiator who controls the private half of the key pair.

## Proof Key Applicability to Multi-Hop Scenarios

In circumstances where end to end signatures are required on information transiting a multi-hop chain, an asymmetric proof key should be employed.

In this case, the SAML token acts as an X.509 certificate equivalent, carrying the public key tied to the identity. There are the added benefits that identity information can be enriched with attributes and that revocation checks are not required due to the SAML Token’s limited lifetime.

In single hop scenarios, either symmetric or asymmetric proof keys may be employed.

## AppliesTo, Audience, and SubjectConfirmation

It is important to note the following when obtaining a SAML token from an Identity Provider:

* Particularly when symmetric keys are employed, SAML tokens need to be targeted at a particular SOAP receiver endpoint. When communicating with an IdP to obtain a suitable token, the SOAP sender must supply an appropriate identifier for the SOAP receiver.
* By convention, this is normally the SOAP endpoint of the receiver, but it may be any URI that logically denotes the SOAP receiver.
* This URI will be placed into the Audience of the SAML token and it will be used by the IdP to determine how to suitably wrap the proof key for consumption by that SOAP Receiver.

For symmetric proof keys, this is essential to preserve the security of the key.

# Error Code Mapping

Introducing SAML as an authentication mechanism adds an additional party into the authentication process and while the nature of the errors does not really change, the locations where they occur does.

Most identity providers will use WS-Trust to issue SAML identity tokens in the web service context although the following mapping applies regardless of the protocol used.

The following table contains the relevant errors to SAML authentication:

|  |  |
| --- | --- |
| EBMS:0005 | ConnectionFailure |
| EBMS:0101 | FailedAuthentication |
| EBMS:0103 | PolicyNoncompliance |

Situations in the token issue and use of SAML tokens will map to the following errors. Not that Issue errors will not be exposed to the SOAP Receiving MSH.

|  |  |  |
| --- | --- | --- |
| **Situation** | **EBMS** | **Type** |
| **0005** | **0101** | **0103** |
| Failure of SOAP Sender to connect to STS  | X |  |  | Issue |
| STS could not authenticate sender |  | X |  | Issue |
| STS could not provide Mandatory claims for Sender |  |  | X | Issue |
| Token signature does not verify |  | X |  | Use |
| Token issuer not recognized by receiver |  | X |  | Use |
| Token Expired |  | X |  | Use |
| Mandatory claims missing from Token |  |  | X | Use |
| Token subject is unknown to Receiver[[1]](#footnote-1) |  |  | X | Use |

# Sample Message

The following example is extracted from the ebMS3 core specification and the X509 Digital Signature is replaced with a SAML based digital signature. Changed regions have been highlighted.

Note: SAML is normally only used for authentication in SOAP Requests. Where Authentication/Non-Repudiation of a SOAP Response is required, it is normal to use an X.509 Signature.

Mime-Version: 1.0

Content-Type: text/xml

Content-Transfer-Encoding: binary

SOAPAction: ""

Content-Length: 7205

<?xml version="1.0" encoding="UTF-8"?>

<S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"

 xmlns:xsd="http://www.w3c.org/2001/XMLSchema"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/

http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms-header-3\_0-200704.xsd">

 <S11:Header xmlns:eb="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/">

 <eb:Messaging id="ebMessage" S11:mustUnderstand="1">

 <eb:UserMessage>

 <eb:MessageInfo>

 <eb:Timestamp>2006-10-31T17:36:20.656Z</eb:Timestamp>

 <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>

 <eb:RefToMessageId>UUID-1@msh-server.example.com</eb:RefToMessageId>

 </eb:MessageInfo>

 <eb:PartyInfo>

 <eb:From>

 <eb:PartyId>uri:msh-server.example.com</eb:PartyId>

 <eb:Role>http://example.org/roles/Buyer</eb:Role>

 </eb:From>

 <eb:To>

 <eb:PartyId type="someType">QRS543</eb:PartyId>

 <eb:Role>http://example.org/roles/Seller</eb:Role>

 </eb:To>

 </eb:PartyInfo>

 <eb:CollaborationInfo>

 <eb:AgreementRef>http://msh-server.example.com/cpa/123456</eb:AgreementRef>

 <eb:Service type="someType">QuoteToCollect</eb:Service>

 <eb:Action>NewPurchaseOrder</eb:Action>

 <eb:ConversationId>2a81ffbd-0d3d-4cbd-8601-d916e0ed2fe2</eb:ConversationId>

 </eb:CollaborationInfo>

 <eb:MessageProperties>

 <eb:Property name="ProcessInst">PurchaseOrder:123456</eb:Property>

 <eb:Property name="ContextID">987654321</eb:Property>

 </eb:MessageProperties>

 <eb:PayloadInfo>

 <eb:PartInfo href="#enc">

 <eb:Description xml:lang="en-US">PO Image</eb:Description>

 </eb:PartInfo>

 </eb:PayloadInfo>

 </eb:UserMessage>

 </eb:Messaging>

 <wsse:Security S11:mustUnderstand="1"

 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"

 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

 <saml:Assertion MajorVersion="1" MinorVersion="1" AssertionID="\_398c2fae-4178-4ff4-ac59-95d491dbbdf6" Issuer="Example Security Token Service" IssueInstant="2013-05-16T02:09:29.645Z" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">

 <saml:Conditions NotBefore="2013-05-16T02:09:29.647Z" NotOnOrAfter="2013-05-16T02:39:29.647Z">

 <saml:AudienceRestrictionCondition>

 <saml:Audience>https://soapreceiver.example.org.au</saml:Audience>

 </saml:AudienceRestrictionCondition>

 </saml:Conditions>

 <saml:AttributeStatement>

 <saml:Subject>

 <saml:NameIdentifier>

 urn:example.org:id:1204567890

 </saml:NameIdentifier>

 <saml:SubjectConfirmation>

 <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key</saml:ConfirmationMethod>

 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

 <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">

 <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">

 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>

 </e:EncryptionMethod>

 <KeyInfo>

 <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

 <X509Data>

 <X509IssuerSerial>

 <X509IssuerName>CN=Example CA, O=Example CA Organisation, C=AU</X509IssuerName>

 <X509SerialNumber>37310890721155718122974868787627716901</X509SerialNumber>

 </X509IssuerSerial>

 </X509Data>

 </o:SecurityTokenReference>

 </KeyInfo>

 <e:CipherData>

 <e:CipherValue>uW2St4BD9+lZzSGbSkvhqIkCoMwVlf3qDJl2X4Nj8bP8LQxJpchQugHKV+7y+8k1vrVxPPapxka7aWscDvGbmHT9cxaAqnNtTuK2R7yo1i22yNxSa3us5l1VHLFB447tAf/tQ/OQPsD4myTqad2+LLoDT6lS0CrJO/Ue+WMLNzI=</e:CipherValue>

 </e:CipherData>

 </e:EncryptedKey>

 </KeyInfo>

 </saml:SubjectConfirmation>

 </saml:Subject>

 <saml:Attribute AttributeName="*A\_Relevant\_Attribute\_or\_Claim*" AttributeNamespace="http://example.org/2008/06/identity/claims">

 <saml:AttributeValue>

 *Relevant Attribute Value*

 </saml:AttributeValue>

 </saml:Attribute>

 </saml:AttributeStatement>

 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

 <SignedInfo>

 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></CanonicalizationMethod>

 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>

 <Reference URI="#\_398c2fae-4178-4ff4-ac59-95d491dbbdf6">

 <Transforms>

 <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></Transform>

 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></Transform>

 </Transforms>

 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>

 <DigestValue>0GEhLfUS2pGSp4ZziJJsV9VbeW8=</DigestValue>

 </Reference>

 </SignedInfo>

 <SignatureValue>DpZX6V4Wn2RI0+a3jug3H5gfa4MZiOGSQ/rfsLHkE0X/HgzV4cZDl4wFtPqBCdm9eyByNtDjzaSKRKT3Md5LgANxMY5deJGJvPmGyQSfrSMrCUCPv5iktaCQEJSpFS+R5KLdSdBkJuFaT6JAYE2CfF6BVk0LGP8LhW/Z6qFfzrA=</SignatureValue>

 <KeyInfo>

 <X509Data>

 <X509Certificate>…</X509Certificate>

 </X509Data>

 </KeyInfo>

 </Signature>

 </saml:Assertion>

 <wsse:BinarySecurityToken

 EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"

 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"

 wsu:Id="encryptionCert">...</wsse:BinarySecurityToken>

 <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">

 <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5"

 xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"/>

 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

 <wsse:SecurityTokenReference>

 <wsse:Reference URI="#encryptionCert"/>

 </wsse:SecurityTokenReference>

 </KeyInfo>

 <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">

 <CipherValue>F3HmZ2Ldyn0umLCx/8Q9B9e8OoslJx9i9hOWQjh6JJwYqDLbdg0QVFiVT1LVjazlThS9m9rkRtpkhCUIY1xjFKtDsuIIAW8cLZv7IHkVoDtQ7ihJc8hYIlEESX9qZN65JgyAa3BYgW9ipjGHtNgZ9RzUdzKdeY74DFm27R6m8b0=</CipherValue>

 </CipherData>

 <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">

 <DataReference URI="#enc"/>

 </ReferenceList>

 </enc:EncryptedKey>

 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

 <ds:SignedInfo>

 <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

 <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>

 <ds:Reference URI="#ebMessage">

 <ds:Transforms>

 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

 </ds:Transforms>

 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

 <ds:DigestValue>Ae0PLUKJUnUyAMXkLQD/WwKiFiI=</ds:DigestValue>

 </ds:Reference>

 <ds:Reference URI="#body">

 <ds:Transforms>

 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

 </ds:Transforms>

 <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

 <ds:DigestValue>kNY6X7LnRTwxXXBzSw07tcA0KSU=</ds:DigestValue>

 </ds:Reference>

 </ds:SignedInfo>

 <ds:SignatureValue>

 T24okA0MUh5iBNMG6tk8QAKZ+lFMmY1rcPnkOr9j3fHRGM2qqUnoBydOTnClcEMzPZbnlhdN

 YZYmab1lqa4N5ynLjwlM4kp0uMip9hapijwL67aBnUeHiFmUau0x9DBOdKZTVa1QQ92106ge

 j2YPDt3VKIlLLT2c8O4TfayGvuY= </ds:SignatureValue>

 <ds:KeyInfo>

 <wsse:SecurityTokenReference

 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" k:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1" xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd">

 <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID">\_398c2fae-4178-4ff4-ac59-95d491dbbdf6</o:KeyIdentifier>

 </wsse:SecurityTokenReference>

 </ds:KeyInfo>

 </ds:Signature>

 </wsse:Security>

 </S11:Header>

 <S11:Body wsu:Id="body"

 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

 <EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"

 xmlns="http://www.w3.org/2001/04/xmlenc#">

 <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>

 <CipherData>

 <CipherValue>tjOgUPMmQwd6hXiHuvl42swqv4dTYiBfmg8u1SuFVRC3yfNlokshvoxs1/qQoqN1prDiSOxsxsFvg1la7dehjMWb0owuvU2de1eKr5KPcSApnG+kTvNrtg==</CipherValue>

 </CipherData>

 </EncryptedData>

 </S11:Body>

</S11:Envelope>

Note in the following example:

* A symmetric proof key has been employed in this example although asymmetric keys could be used if required.
* When symmetric keys are used, the must be encrypted for the receiver.
* Only a single attribute/claim and value have been included for illustrative purposes, but zero or more may be provided in the general case.
* The SAML Token has replaced the BinarySecurityToken for the X.509 Certificate for the signer.
* The signing algorithm has been changed to reflect the symmetric signing key used.
* The SecurityTokenReference in the message signature has been changed to refer to the SAML token.
* Encryption is unchanged.

# Processing Modes for SAML Security

The following PModes are specific to a SAML implementation. These PModes are specific to the SOAP receiver and indicate what the SOAP sender requires in order to communicate with that receiver. These PModes may be expressed in accordance with [WSPOLICY] and [WSSECPOLICY] for both WSDL and MEX, but that is dependent on the implementation.

PMode[1].Security.SAML.Version: The value of this parameter is a list of the versions of SAML Tokens supported. At present, this will be either SAML11, SAML20, or both.

PMode[1].Security.SAML.IdP: The value of this parameter is a list of the URLs of IdPs acceptable to the SOAP receiver. Each IdP URL may additionally have an associated URI by which the IdP knows the SOAP receiver. If absent, the default should be used. (This is normally the URL of the receiver endpoint.)

PMode[1].Security.SAML.MandatoryAttributes: The value of this parameter is a list of SAML Attributes describing the subject that are required in the SAML Assertion.

PMode[1].Security.SAML.OptionalAttributes: The value of this parameter is a list of SAML Attributes describing the subject that should be provided in the SAML Assertion, but are not required.

PMode[1].Security.SAML.KeyType: The value of this parameter denotes the type of proof key require in the SAML Assertion. The key type may be Symmetric or Asymmetric.

The following PModes are unchanged from [EBMS3CORE] and are apply to SAML in the same way they apply to X.509:

PMode[1].Security.X509.Signature.HashFunction: The value of this parameter identifies the algorithm that is used to compute the digest of the message being signed. The definitions for these values are in the [XMLDSIG] specification.

PMode[1].Security.X509.Signature.Algorithm: The value of this parameter identifies the algorithm that is used to compute the value of the digital signature. The definitions for these values are found in the [XMLDSIG] or [XMLENC] specifications.

Encryption of elements should use the X.509 PModes.

# How is SAML Beneficial [Informational/Non-Normative]

SAML is oft presented as being more complex than X.509 and Username/Password. Mostly, it is that it is newer and hence less well supported and understood. It is more powerful and has long term benefits that will become apparent in this document.

SAML can be used in small fixed communities, but where SAML comes in to its own is in *many to few* scenarios. SAML is particularly suited to *client polling hub* scenarios.

While SAML is normally used for securing SOAP requests, X.509 is normally used for securing the SOAP responses.

## What is a SAML Token?

A SAML Token is a relatively small XML document fragment which certifies the identity and attributes of an entity transaction on the internet.

In common scenarios, this entity may be a person, a business representative, or a business back end system.



SAML Tokens are normally ephemeral. They convey identity for a short period of minutes to hours. They can be used in a number of contexts, from web based single sign on to web services.

A SAML token normally contains a subject which will be a unique identifier for the entity. It is typically used for tracking and authorisation purposes where the SOAP receiver has its own authorisation database.

A SAML token also contains attributes of the entity. In WS-Trust parlance, these attributes are referred to as claims. A claim has a name and a value. Claims typically contain verified information about the entity such as email address. They could contain information such as the DUNS number of a business entity or other common registration number for a particular community.

SAML attributes can also contain role information for an entity. A business representative may carry a claim of “Authorised Purchasing Officer” for example.

The SAML token has an XML-DSIG signature of a trusted identity provider covering it to certify the validity and correctness of the information contained in the token.

When SAML tokens are used in web services, they have a cryptographic key, the Proof Key, which binds the identity of the owner to messages they are sending in the same way that an X.509 Certificate does.

Unlike an X.509 Certificate, because SAML tokens are normally relatively short lived and are populated with up to date information at the time they are issued (typically within a few minutes of performing a transaction), there is no need for a revocation mechanism.

Based on the sensitivity of a transaction, a token receiver can always reject a token based on authentication event that the token was based on being outside its tolerance.

## Obtaining a SAML Token

In web services scenarios, SAML Tokens are provided as they are required from an Identity Provider, typically using the WS-Trust protocol. In WS-Trust parlance, the Identity Provider is referred to as a Security Token Service (STS)

The following diagram shows a typical flow for a WS-Trust based identity provider:



Prior to any transactions taking place a small amount of registration needs to take place. For the simple case where only one identity provider involved, it will involve something like the following:

1. The Initiating MSH is configured with a credential that has been obtained from an Identity Provider. Obtaining such a credential typically requires some sort of Evidence of Identity for the registrant and a Proof of Association with the business.
Typically validated attributes associated with the credential will be held by the credential provider. (For example,a unique business identifier.)
2. The Responding MSH registers with the Identity Provider as a relying party. At a minimum, the relying party registration requires:
	* An endpoint for the service they are hosting
	* An X.509 Certificate that can be used for wrapping proof keys so that only the relying party can extract them.

Optionally a set of required attributes (claims) may be registered.
In return, the Identity Provider provides an X.509 Certificate that will be used to verify the siugnatures on SAML Assertions (SAML Tokens) supplied by the Identity Provider.

At transmission time, the following steps would normally be taken:

1. The Initiating MSH makes a WS-Trust call to the Identity Provider in the form of a Request Ssecurity Token message. The call includes:
	* The endpoint that the Initiating MSH is targeting. This is supplied in the “AppliesTo” element of the request. (the Responding MSH)
	* Authentication information that verifies the identity of the Initiating MSH to the Identity Provider. (Typically through a WS-Security compliant mechanism.)
	* A list of the attributes (claims) that the Initiating MSH wishes to have asserted to the Responding MSH.
2. The Identity Provider responds with a Request for Security Token Response message which includes:
	* A SAML Assertion (SAML Token) including:
		+ - the identity of the Initiating MSH,
			- the required attributes (claims),
			- the key for verifying signatures made by the Initiating MSH, encrypted for the Responding MSH
	* A Proof Key to be used by the Initiating MSH to sign requests to the Responding MSH.
3. The Initiating MSH makes a SOAP call to the Responding MSH including:
	* The standard ebMS3 SOAP headers appropriate for the request
	* The WS-Security header including:
		+ The message signature, signed with the Proof Key from step 2;
		+ A reference to the SAML Token in the signature’s KeyInfo
		+ The SAML Token
	* The standard ebMS3 SOAP body
4. The Responding MSH returns a SOAP response to the Sending MSH as it would normally in an X.509 scenario, signing the response with its X.509 Certificate.

Note that SAML tokens are typically re-used for a period of time, normally up to 30 minutes, so steps 1 and 2 may be followed by multiple 3 and 4 pairs until the SAML token expires.

## Usage Scenarios

## Independent Identity Provider (STS)



In this scenario, the Identity Provider is an independent third party. They have means of establishing the Sender’s identity, obtaining the required attributes and preparing a SAML token that can be used with the Receiver.

The Receiver has established a one off setup with the Identity Provider, exchanging credentials so that the Receiver can validate tokens from the Identity Provider and the Identity Provider can encrypt verification privately for the receiver that may be included in the SAML token.

## Key Benefits

* The Identity Provider handles all aspects of identifying the user including issues around credential loss or rollover.
* The Receiver only needs to accept one type of credential, the SAML token. The Sender can authenticate with any type of credential supported by the Identity Provider provided the required claims are available.

## Receiver End Authorisation Provider (STS)



The Identity Provider can live in the receiver’s domain. Typically this approach is employed when there is a need to share authorisation information across systems and/or technologies.

## Key Benefits

* Identity and Role information are available in web service context without access to external systems or requiring external systems to populate a repository in the MSH.
* Role information is maintained separately and is accessible and consistent across channels.

## Sender End Authorisation Provider (STS)



In business models where the Sender is in a desktop application rather than a back end system, individual users in a business will have various authorities to represent that business.

In this case, the Sender will contact an Identity Provider inside their own business for a SAML token. This will be transparent to the user, but will be performed by the domain authentication system. The issued SAML token will be created by the STS with the appropriate claims to indicate the user’s authority to represent the business.

## Key Benefits

* No additional credentials for the users to keep (usernames or certificates) in order transact with external parties.
* Business manages its own authorisations rather than having to rely on external services to stop its users exceeding their authority.

## Chained Identity Provider Model

Identity Providers can be chained. That is, one SAML token can be used to obtain another.



In this model, the Sender obtains a series of tokens:

Their domain logon gets them a token that flags them as a valid organisation representative with a set of established roles through its issued SAML token.

The organisation SAML token is presented to the Independent Identity Provider which establishes the organisations identity and passes through claims that can be legitimately asserted by the organisation as well as enriching the SAML token with any required claims that it may have.

The Receiver Identity Provider then issues a SAML token based on the information provided by the Independent Identity Provider, enriching it as appropriate with information that it holds about the business and/or its representative.

## Key Benefits

* Organisations can control what external roles their representatives are entitled to.
* Sending organisations only need to be registered with one Identity Provider and not with each Receiver.
* Receivers can override or augment identity/role information in a single place before it enters internal systems.
* Identity information can be shared between systems using a standards based mechanism.
1. Just because an Identity Provider can authenticate an entity, that does not mean that the entity is registered with the SOAP receiver for the purposes of authorisation and access. [↑](#footnote-ref-1)